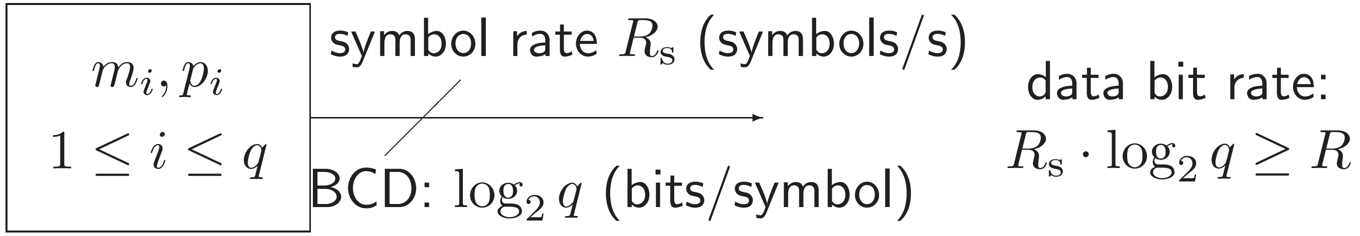
Revision of Lecture 1

- Source emitting sequence $\{S(k)\}$: 1. Symbol set m_i, $1 \le i \le q$; 2. Probability of occurring for m_i, p_i, $1 \le i \le q$; 3. Symbol rate R_b [symbols/s]; 4. **Memoryless**



- **Information content** $I(m_i) = \log_2 \frac{1}{p_i}$ (bits)

  Entropy $H = \sum_{i=1}^{q} p_i \log_2 \frac{1}{p_i}$ (bits/symbol)

  Information rate $R = R_s \cdot H$ (bits/s)

- After source coding, e.g. code every symbol by $\log_2 q$ bits or binary coded decimal (BCD), data rate you send is R_b [bits/s] and generally $R_b > R$

- How to code symbols to achieve **efficiency** (data bit rate $R_b = R$)?

Maximum Entropy for q-ary Source

- Entropy of a q-ary source: $H = -\sum_{i=1}^{q} p_i \log_2 p_i$ when it reaches maximum?

- Maximisation under the constraint $\sum_{i=1}^{q} p_i = 1$ is based on the Lagrangian

$$\mathcal{L} = \left(\sum_{i=1}^{q} -p_i \log_2 p_i \right) + \lambda \cdot \left(1 - \sum_{i=1}^{q} p_i \right)$$

  and yields

$$\frac{\partial \mathcal{L}}{\partial p_i} = -\log_2 p_i - \log_2 e - \lambda = 0$$

- Since $\log_2 p_i = -(\log_2 e + \lambda)$ is independent of i, i.e. constant, and $\sum_{i=1}^{q} p_i = 1$, entropy of a q-ary source is maximised for equiprobable symbols with $p_i = 1/q$

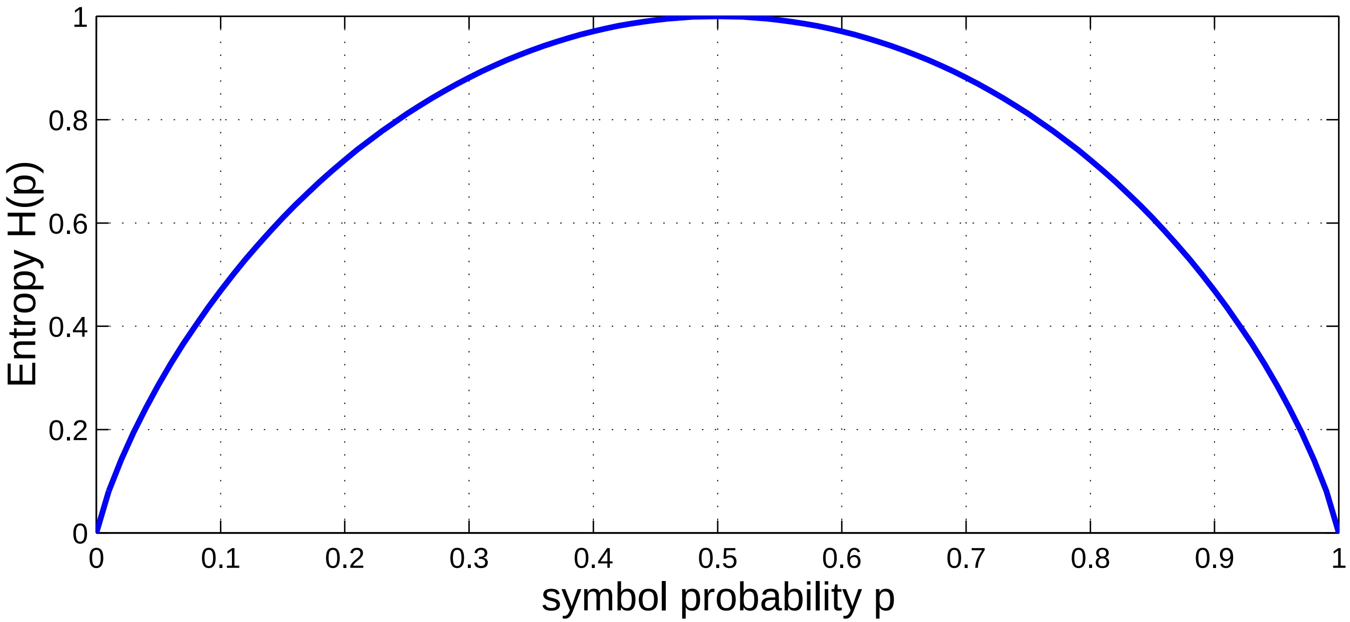
$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \rightarrow 1 = \sum_{i=1}^{q} p_i, \ \text{also } p_i = c: \quad 1 = \sum_{i=1}^{q} c \rightarrow p_i = c = \frac{1}{q}$$

Maximum Entropy for binary Source

Binary source ($q = 2$) emitting two symbols with probabilities $p_1 = p$ and $p_2 = (1-p)$:

- Source entropy:

$$H(p) = -p \cdot \log_2 p - (1-p) \cdot \log_2(1-p)$$



- Source entropy is maximum for equiprobable symbols, $p_1 = p_2 = 0.5$

Efficient Source Coding

- We are considering **lossless** source coding, i.e. when we convert symbols into bit streams (codewords), we do not throw away any "information"

- Information rate is $R = R_{\mathrm{s}} \cdot H \leq R_{\mathrm{s}} \cdot \log_2 q$, so in general BCD is not efficient

- $0 \leq H \leq \log_2 q$, so source entropy is bound by maximum entropy and, therefore, BCD only achieves most efficient source coding for **equiprobable** symbols

- Efficient channel use requires efficient source encoding, and coding efficiency:

$$\text{coding efficiency} = \frac{\text{source information rate}}{\text{average source output rate}}$$

- Shannon's source coding theorem: **with an efficient source coding, a coding efficiency of almost 100% can be achieved**

- We consider efficient Shannon-Fano and Huffman source codings

Efficient Coding (continue)

- For source with equiprobable symbols, it is easy to achieve an efficient coding
  - For such a source, $p_i = 1/q$, $1 \le i \le q$, and source entropy is maximised: $H = \log_2 q$ bits/symbol
  - Coding each symbol into $\log_2 q$-bits codeword is efficient, since coding efficiency[2]

$$\text{CE} = \frac{H}{\log_2 q} = \frac{\log_2 q}{\log_2 q} = 100\%$$

- For source with non-equiprobable symbols, coding each symbol into $\log_2 q$-bits codeword is not efficient, as $H < \log_2 q$ and

$$\text{CE} = \frac{H}{\log_2 q} < 100\%$$

- How to be efficient: **assign number of bits to a symbol according to its information content**, that is, using **variable-bits codewords**, more likely symbol having fewer bits for its codeword

[2] Assume $\log_2 q$ is an integer, as practical coding cannot have a fraction of bit

Example of 8 Symbols

symbol	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
BCD codeword	000	001	010	011	100	101	110	111
equal p_i	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
non-equal p_i	0.27	0.20	0.17	0.16	0.06	0.06	0.04	0.04

Average codeword length is 3 bits for BCD

- For equiprobable case, $H = \log_2 8 = 3$ bits/symbol, and coding each symbol with 3-bits codeword achieves coding efficiency of 100%

- For non-equiprobable case,

$$\begin{aligned} H &= -0.27\log_2 0.27 - 0.20\log_2 0.20 - 0.17\log_2 0.17 - 0.16\log_2 0.16 \\ &\quad -2\times 0.06\log_2 0.06 - 2\times 0.04\log_2 0.04 = 2.6906 \quad \text{(bits/symbol)} \end{aligned}$$

coding each symbol by 3-bits codeword has coding efficiency

$$\text{CE} = 2.6906/3 = 89.69\%$$

Shannon-Fano Coding

Shannon-Fano source encoding follows the steps

1. Order symbols m_i in descending order of probability
2. Divide symbols into subgroups such that the subgroup's probabilities (**i.e. information contests**) are as close as possible

   can be two symbols as a subgroup if there are two close probabilities (i.e. information contests), can also be only one symbol as a subgroup if none of the probabilities are close
3. Allocating codewords: assign bit 0 to top subgroup and bit 1 to bottom subgroup
4. Iterate steps 2 and 3 as long as there is more than one symbol in any subgroup
5. Extract variable-length codewords from the resulting tree (*top-down*)

Note: Codewords must meet condition: no codeword forms a *prefix* for any other codeword, so they can be decoded *unambiguously*

Shannon-Fano Coding Example

- Example for 8 symbols

approx length	I_i (bits)	Symb. m_i	Prob. p_i	Coding Steps 1	2	3	4	Codeword
2	1.89	m_1	0.27	0	0			00
2	2.32	m_2	0.20	0	1			01
3	2.56	m_3	0.17	1	0	0		100
3	2.64	m_4	0.16	1	0	1		101
4	4.06	m_5	0.06	1	1	0	0	1100
4	4.06	m_6	0.06	1	1	0	1	1101
4	4.64	m_7	0.04	1	1	1	0	1110
4	4.64	m_8	0.04	1	1	1	1	1111

- Less probable symbols are coded by longer code words, while higher probable symbols are assigned short codes

Assign number of bits to a symbol as close as possible to its information content, and no codeword forms a prefix for any other codeword

Shannon-Fano Coding Example (continue)

- Entropy for the given set of symbols: $H = 2.6906$ (bits/symbol)
- Average code word length with Shannon-Fano coding:

$$0.47 \cdot 2 + 0.33 \cdot 3 + 0.2 \cdot 4 = 2.73 \quad \text{(bits/symbol)}$$

  Coding efficiency:

$$\frac{\text{source information rate}}{\text{average source output rate}} = \frac{R_s \cdot H}{R_s \cdot 2.73} = \frac{2.6906}{2.73} = 98.56\%$$

- In comparison, coding symbols with 3-bits equal-length codewords:
  - Average code word length is 3 (bits/symbol)
  - Coding efficiency is 89.69%

Shannon-Fano Coding – Another Example

Symbol X_i	Prob. $P(X_i)$	I (bits)	Codeword							bits/symbol
A	$\frac{1}{2}$	1	0							1
B	$\frac{1}{4}$	2	1	0						2
C	$\frac{1}{8}$	3	1	1	0					3
D	$\frac{1}{16}$	4	1	1	1	0				4
E	$\frac{1}{32}$	5	1	1	1	1	0			5
F	$\frac{1}{64}$	6	1	1	1	1	1	0		6
G	$\frac{1}{128}$	7	1	1	1	1	1	1	0	7
H	$\frac{1}{128}$	7	1	1	1	1	1	1	1	7

Source entropy

$$H = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{1}{16}\cdot 4 + \frac{1}{32}\cdot 5 + \frac{1}{64}\cdot 6 + 2\cdot\frac{1}{128}\cdot 7 = \frac{127}{64} \quad \text{(bits/symbol)}$$

Average bits per symbol of Shannon-Fano coding

$$\frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{1}{16}\cdot 4 + \frac{1}{32}\cdot 5 + \frac{1}{64}\cdot 6 + 2\cdot\frac{1}{128}\cdot 7 = \frac{127}{64} \quad \text{(bits/symbol)}$$

Coding efficiency is 100% (Coding efficiency is 66% if codewords of equal length of 3-bits are used)

Huffman Coding

Huffman source encoding follows the steps

1. Arrange symbols in descending order of probabilities
2. Merge the two least probable symbols (or subgroups) into one subgroup
3. Assign '0' and '1' to the higher and less probable branches, respectively, in the subgroup
4. If there is more than one symbol (or subgroup) left, return to step 2
5. Extract the Huffman code words from the different branches (*bottom-up*)

Huffman Coding Example

- Example for 8 symbols

Symb. m_i	Prob. p_i	Coding Steps 1	2	3	4	5	6	7	Code word
m_1	0.27						1	0	01
m_2	0.20					0		1	10
m_3	0.17				0		0	0	000
m_4	0.16				1		0	0	001
m_5	0.06		0	0		1		1	1100
m_6	0.06		1	0		1		1	1101
m_7	0.04	0		1		1		1	1110
m_8	0.04	1		1		1		1	1111

- Intermediate probabilities: $m_{7,8} = 0.08$; $m_{5,6} = 0.12$; $m_{5,6,7,8} = 0.2$; $m_{3,4} = 0.33$; $m_{2,5,6,7,8} = 0.4$; $S_{1,3,4} = 0.6$

- When extracting codewords, remember "**reverse bit order**" - This is important as it ensures no codeword forms a prefix for any other codeword

Huffman Coding Example (explained)

step 2

m_1	0.27	
m_2	0.20	
m_3	0.17	
m_4	0.16	
m_{78}	0.08	
m_5	0.06	0
m_6	0.06	1

step 3

m_1	0.27	
m_2	0.20	
m_3	0.17	
m_4	0.16	
m_{56}	0.12	0
m_{78}	0.08	1

step 4

m_1	0.27	
m_2	0.20	
m_{5678}	0.20	
m_3	0.17	0
m_4	0.16	1

step 5

m_{34}	0.33	
m_1	0.27	
m_2	0.20	0
m_{5678}	0.20	1

step 6

m_{25678}	0.40	
m_{34}	0.33	0
m_1	0.27	1

step 7

m_{134}	0.60	0
m_{25678}	0.40	1

- Average code word length with Huffman coding for the given example is also 2.73 (bits/symbol), and coding efficiency is also 98.56%
- Try Huffman coding for 2nd example and compare with result of Shannon-Fano coding

Shannon-Fano and Huffman Source Encoding Summary

- Shannon-Fano and Huffman source encoding methods belong to class of **entropy encoding**, which
  - Assign number of bits to a symbol as close as possible to its information content
- Both Shannon-Fano and Huffman coded sequences can be *decoded unambiguously*, as no code word forms a prefix for any other code word
- With Shannon-Fano and Huffman coding, *memory-less sources* can be encoded such that the emitted bit sequence carries a *maximum of information*
- For an alphabet of q symbols, the longest code word could be up to $q-1$ bits; this is *prohibitive for large alphabets* (requiring large buffer sizes)
- Huffman coding gives a different code word assignment to Shannon-Fano coding; the *coding efficiency* is however *nearly identical*
- Which of these two source encoding methods do you prefer?

Summary

- Source encoding efficiency and concept of **efficient encoding**
- Memoryless source with equiprobable symbols achieves maximum entropy
- Entropy source encoding: Shannon-Fano and Huffman source encoding methods
  - What for: **Data rate** $>$ **information rate** $\rightarrow$ **With efficient encoding, data rate is minimised, i.e. as close to information rate as possible**
  - How to: Assign number of bits to a symbol as close as possible to its information content, subject to practical constraints of
    1. Number of bits can only be an integer
    2. No code word forms a prefix for any other code word